CSS Mastery

Advanced Web Standards Solutions, Third Edition

精通CSS

高级Web标准解决方案

（第3版）

[英] 安迪·巴德 [瑞典] 埃米尔·比约克隆德 著

李松峰 译

- CSS畅销经典全面升级，充分展示现代CSS实践技巧
- 直接提供常见问题的解决方案，让前端架构更上一层楼

中国工信出版集团 人民邮电出版社 POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

4P5.

CSS Mastery

Advanced Web Standards Solutions, Third Edition

精通CSS

高级Web标准解决方案

（第3版）

[英] 安迪·巴德 [瑞典] 埃米尔·比约克隆德 著

李松峰 译

人民邮电出版社

北 京

图书在版编目（CIP）数据

精通CSS : 高级Web标准解决方案 : 第3版 / (英)安迪·巴德 (Andy Budd) , (瑞典) 埃米尔·比约克隆德 (Emil Björklund) 著 ; 李松峰译. -- 北京 : 人民邮电出版社, 2019.2
（图灵程序设计丛书）
ISBN 978-7-115-50690-0

Ⅰ. ①精… Ⅱ. ①安… ②埃… ③李… Ⅲ. ①网页制作工具 Ⅳ. ①TP393.092.2

中国版本图书馆CIP数据核字(2019)第020503号

内 容 提 要

本书是CSS设计经典图书升级版，结合CSS近年来的发展，尤其是CSS3和HTML5的特性，对内容进行了全面改写。本书介绍了涉及字体、网页布局、响应式Web设计、表单、动画等方面的实用技巧，并讨论了如何实现稳健、灵活、无障碍访问的Web设计，以及在技术层面如何实现跨浏览器方案和后备方案。本书还介绍了一些鲜为人知的高级技巧，让你的Web设计脱颖而出。

本书适合具备HTML和CSS基础知识的读者阅读。

◆ 著　[英] 安迪·巴德
　　[瑞典] 埃米尔·比约克隆德
　译　李松峰
　责任编辑　温　雪
　责任印制　周昇亮
◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路11号
　邮编　100164　电子邮件　315@ptpress.com.cn
　网址　http://www.ptpress.com.cn
　涿州市京南印刷厂印刷
◆ 开本：800×1000　1/16
　印张：24.25
　字数：573千字　2019年2月第1版
　印数：1-4 000册　2019年2月河北第1次印刷
著作权合同登记号　图字：01-2016-8299号

定价：99.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字20170147号

版权声明

本书献给我在 Clearleft 的同事——过去的，还有现在的。没有他们的支持和智慧，就没有这本书。

——安迪·巴德（Andy Budd）

献给我最怀念的祖父 Sven Forsberg（1919—2016），一位工程师、艺术家、终身手艺人。

——埃米尔·比约克隆德（Emil Björklund）

前　言

回想 2004 年，在写本书第 1 版的时候，市面上已经有两本 CSS 方面的书了。当时，我对读者是否需要第三本并没有把握。毕竟，CSS 那时候还算小众技术，只有博主和 Web 标准的狂热粉丝才会研究。当时的大部分网站用的还是表格和框架。本地开发者邮件列表中的小伙伴都说我疯了，他们认为 CSS 只是一个美丽的梦。他们并没有想到，那时候 Web 标准运动的序幕即将拉开，而本书的出版恰逢这个领域爆发之际。在接下来的几年里，本书一直名列出版商最畅销图书榜单。

等到本书第 2 版出来的时候，CSS 的地位已经无法撼动。本书的作用也从向人们展示 CSS 的魅力，转变成帮人们更有效地使用 CSS。于是，我们找到各种新技术、解决方案，还有“黑科技”，希望打造出一本 Web 设计师和前端开发者的权威指南。当时 CSS 的发展似乎已经趋于稳定，而本书好像也能卖相当长一段时间。事实证明，我们错了。

CSS 的发展并未停滞，近几年的情况表明，CSS 最终兑现了其最初的承诺。我们进入了 Web 标准的黄金时代，即浏览器支持程度已经足够好的时代。因此，我们终于可以放弃原来那些“黑科技”，转而把时间和精力花在为最大、最复杂的网站编写优雅、巧妙、容易维护的代码上。

于是，本书第 3 版应运而生：该把所有新工具、新技术和新思路写成一本新书了。为了完成这个任务，我把好朋友埃米尔·比约克隆德拉了进来。他是一位技术与才能俱佳的开发高手，为本书加入了对现代 CSS 实践的深刻理解，还会告诉大家怎么利用新技术写出高度灵活的代码，并且让这些代码以最优雅的方式在不同浏览器、不同屏幕和不同平台上跑起来。

应该说，我们俩通力合作，基本上完全重写了本书，并且添加了覆盖 Web 排版、动画、布局、响应式设计、组织代码等主题的新章节。这一版仍然继承了前两版的写法，整合了实例、语言解读和跨浏览器的巧妙解决方案。谙熟各路“黑科技”或者任意属性都能信手拈来，这些不再是精通 CSS 的标志。今天的 CSS 已经分化为几十个规范，演化出了几百个属性，恐怕没有谁能够对其无所不知！因此，这一版不追求让读者对 CSS 无所不知，而是强调灵活性、稳健性，并确保代码在花样不断翻新的浏览器、设备和使用场景下都能欢快地跑起来。虽然本书不会一一介绍所有语言特性，但会让你知道有什么可用，告诉你一些鲜为人知的基本技术，还有对 CSS 未来的展望。

要想真正看懂这本书，读者至少应该懂得 CSS 的基本原理，比如自己写过 CSS，甚至用它

设计过一两个网站。本书前三章是科普性质的，讲了一些给网页添加样式的最基础的知识，也算是照顾一下基础不牢的读者吧。从第 4 章开始，每一章都会介绍不同的 CSS 新特性，给出的例子也会越来越复杂。相信即使是 CSS 的老手，也能从本书中学到解决常见问题的实用技术。当然，这样的读者就不用按部就班地从头看到尾了，感觉哪一章有意思就看哪一章吧。

最后，我们希望无论读者的基础如何、经验多寡，都能够借助本书领略到 CSS 的无穷魅力，最终成为真正精通 CSS 的大师。

本书源代码地址：https://github.com/Apress/css-mastery-16。[①]

① 读者可前往本书图灵社区页面（http://www.ituring.com.cn/book/1910）下载源代码，查看本书更多信息，并提交中文版勘误。——编者注

致　谢

我们要感谢 Jeffrey Zeldman、Eric Meyer 和 Tantek Çelik 的不懈努力，如果没有他们，“Web 标准运动”就永远不会发生。我们还要感谢后来参与这场运动的人，比如 John Allsopp、Rachel Andrew、Mark Boulton、Doug Bowman、Dan Cederholm、Andy Clarke、Simon Collison、Jon Hicks、Molly E. Holzschlag、Aaron Gustafson、Shaun Inman、Jeremy Keith、Peter-Paul Koch、Ethan Marcotte、Drew McLellan、Cameron Moll、Dave Shea、Nicole Sullivan 和 Jason Santa-Maria，他们迎接挑战并齐心协力将 CSS 变成今天的主流。最后，我们要感谢所有坚持不懈的设计师和开发者，他们接过了接力棒，并让 CSS 成为我们今天所知道的现代设计语言。虽然难免挂一漏万，但这里还是要提几位近年来对我们的实践产生了极大影响的人，包括 Chris Coyier、Vasilis van Gemert、Stephen Hay、Val Head、Paul Lewis、Rachel Nabors、Harry Roberts、Lea Verou、Ryan Seddon、Jen Simmons、Sara Soueidan、Trent Walton 和 Estelle Weyl。我们还要感谢那些在 Twitter 和 Slack 小组中给过我们帮助和启发的设计师和开发者。

我们要感谢帮助这本书冲过终点的每一个人，包括为写作本书提供赞助的 inUse 公司。特别感谢技术编辑 Anna Debenham，如果书中存在任何错误，那么一定是我们没搞好，并且导致她没有发现。我们还要感谢 Andy Hume，他在本书开始写作时提供了专业的意见，为这个新版本确定了方向。此外，我们要感谢 Charlotte Jackson、Peter-Paul Koch、Paul Lloyd、Mark Perkins 和 Richard Rutter，感谢他们审校本书草稿、贡献想法，并提供宝贵的反馈。

书中或示例中的照片多数是我们自己制作或者从公共领域搜集的。以下图片获得了 Creative Commons Attribution 2.0 许可授权：*Portrait*，由 Jeremy Keith 提供；*A Long Night Falls on Saturn's Rings*，由美国国家航空和航天局戈达德太空飞行中心提供。

最后，我们俩都想感谢各自的伴侣，写那么多页花了太长时间，感谢她们的耐心和支持。

目　录

第 1 章 基础知识

人类天生好奇，喜欢摆弄东西。那天在办公室收到了新的遥控四轴飞行器 Parrot AR Drone，我们都没看说明书，就开始动手组装起来。我们喜欢自己琢磨，喜欢按照自己认为的事物运作方式来思考和解决问题。零件只要能拼上就行，除非拼不上了，或者事实跟我们的想法相背离，才不得已去翻翻安装指南。

学习层叠样式表（CSS，cascading style sheets）最好的方法也一样，就是不管三七二十一，直接上手写代码。事实上，可能很多人都是这么学习编程的。比如在某个博客上看到了一些建议，又比如通过开源代码库研究自己喜欢的设计师创造的某个特效的源代码。几乎没人是在完整地看了一遍规范全文之后才开始动手写代码的，因为那样的话，人也早睡着了。

自己动手写代码是最好的开始方式，只是如果不够细心，可能会误解某个重要概念，或者给以后写代码埋下隐患。我们对此深有体会，因为自己给自己挖坑的事儿已经干了不是一回两回了。因此，这一章就来回顾一些基础但容易误解的概念，讲一讲怎么让 HTML 和 CSS 保持组织分明且结构良好。

本章内容：

- 可维护性的重要意义
- HTML 和 CSS 的不同版本
- 未来友好的代码与向后兼容的代码
- 使用新的 HTML5 元素为 HTML 赋予意义
- 在 HTML 中为添加样式设置恰当的接入点
- 通过 ARIA、微格式、微数据扩展 HTML 的语义
- 浏览器引擎模式与验证

1.1 组织代码

人们通常不会注意到建筑物的基础。可是，没有坚实的基础，建筑物的主体就不可能稳固。虽然本书讲的是 CSS 的技术和概念，但这些内容的讲解与实现必须以结构良好且有效的 HTML 文档为前提。

本节会介绍结构良好且有意义的 HTML 文档对基于标准的 Web 开发的重要性，以及如何让文档更有意义、更灵活，从而减轻开发者的工作负担。但首先我们要讲一个对任何语言都同样重要的概念。

1.1.1　可维护性

可维护性可以说是所有优秀代码最重要的特点。如果你的代码已经看不出结构，变得难以阅读，那么很多问题就会接踵而至。开发新功能、修复 bug、提升性能，所有这些操作都会因为代码可读性差以及代码脆弱而变得复杂，而且结果难料。这最终会导致开发人员一点代码也不敢改，因为每次只要改一点，就会出问题。于是，没人愿意再维护这个网站，更糟糕的情况是只能走严格的变更控制流程，每周发布一次，甚至每月才能发布一次！

如果你开发的网站最终要交付给客户或者另一个开发团队，那么可维护性就更重要了。这时候，代码是否容易看懂，是否意图明确，是否**为将来的修改做过优化**，都至关重要。哪个项目没有持续不断的变更需求？哪个项目不需要一直开发新功能？哪个项目不需要不断修复 bug？因此，“唯一不变的就是变化”。

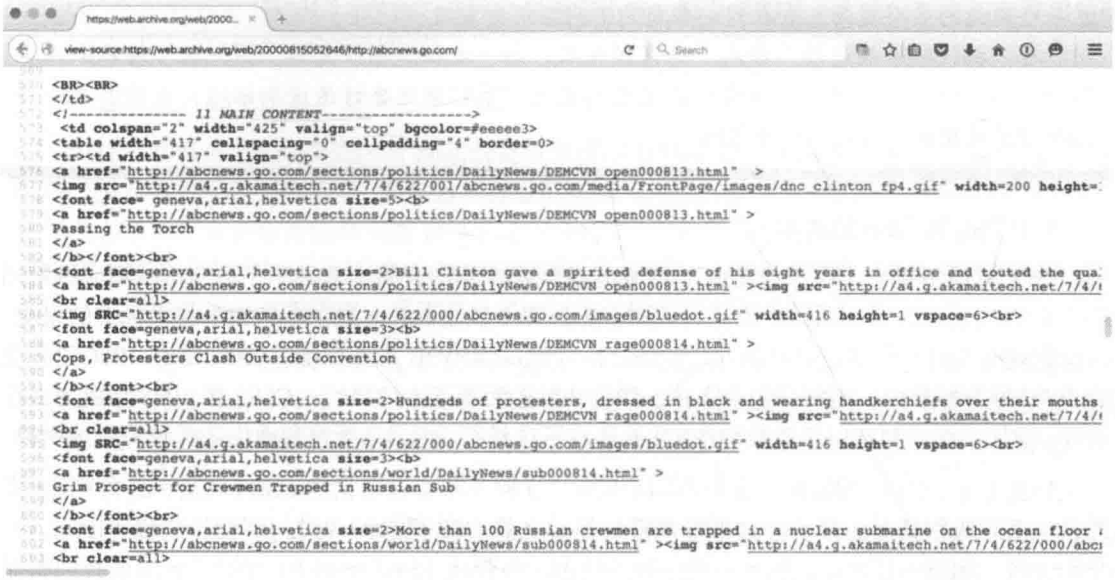
相对来说，CSS 是随着代码量增加而最难保持可维护性的语言之一。即使网站规模不大，样式表也会很快变得难以控制。现代编程语言都有内置的变量、函数和命名空间等特性，这些特性都有利于保持代码的结构和模块化。这些特性 CSS 都没有，所以要按照使用这种语言和组织代码的特殊方式来管理它。本书后面在讨论各种主题时，大都会涉及可维护性。

1.1.2　HTML 简史

> Web 的威力源自其普适性。即使残障用户也能使用是题中应有之义。
>
> ——Tim Berners-Lee

Tim Berners-Lee 在 1990 年发明了 HTML，当时是为了规范科研文档的格式。HTML 是一种简单的标记语言，为文本赋予了基本的结构和意义，比如标题、列表、定义等。这些文档通常没有什么装饰性的元素，可以方便地通过计算机来检索，而人类可以使用文本终端、Web 浏览器，或者必要时使用屏幕阅读器来阅读它们。

然而，人类是视觉发达的生物。随着万维网被越来越多的人所接受，HTML 也逐渐增加了对展示效果的支持。除了用标题元素标记文档标题，还可以使用粗体标签和不同的字体来创建特殊的视觉效果。本来用于展示数据的表格（`table`），却成了页面布局的手段；块引用（`blockquote`）也经常被用来缩进文本，而不是只用来标记引文。HTML 很快就偏离了为内容赋予结构和意义的初衷，变成了一堆字体和表格标签。Web 设计者给这种标记起了个名字，叫“标签汤”（见图 1-1）。



```
<BR><BR>
</td>
<!-------------- II MAIN CONTENT------------------->
 <td colspan="2" width="425" valign="top" bgcolor=#eeeee3>
<table width="417" cellspacing="0" cellpadding="4" border=0>
<tr><td width="417" valign="top">
<a href="http://abcnews.go.com/sections/politics/DailyNews/DEMCVN_open000813.html" >
<img src="http://a4.g.akamaitech.net/7/4/622/001/abcnews.go.com/media/FrontPage/images/dnc_clinton_fp4.gif" width=200 height=
<font face= geneva,arial,helvetica size=5><b>
<a href="http://abcnews.go.com/sections/politics/DailyNews/DEMCVN_open000813.html" >
Passing the Torch
</a>
</b></font><br>
<font face=geneva,arial,helvetica size=2>Bill Clinton gave a spirited defense of his eight years in office and touted the qua
<a href="http://abcnews.go.com/sections/politics/DailyNews/DEMCVN_open000813.html" ><img src="http://a4.g.akamaitech.net/7/4/
<br clear=all>
<img SRC="http://a4.g.akamaitech.net/7/4/622/000/abcnews.go.com/images/bluedot.gif" width=416 height=1 vspace=6><br>
<font face=geneva,arial,helvetica size=3><b>
<a href="http://abcnews.go.com/sections/politics/DailyNews/DEMCVN_rage000814.html" >
Cops, Protesters Clash Outside Convention
</a>
</b></font><br>
<font face=geneva,arial,helvetica size=2>Hundreds of protesters, dressed in black and wearing handkerchiefs over their mouths
<a href="http://abcnews.go.com/sections/politics/DailyNews/DEMCVN_rage000814.html" ><img src="http://a4.g.akamaitech.net/7/4/
<br clear=all>
<img SRC="http://a4.g.akamaitech.net/7/4/622/000/abcnews.go.com/images/bluedot.gif" width=416 height=1 vspace=6><br>
<font face=geneva,arial,helvetica size=3><b>
<a href="http://abcnews.go.com/sections/world/DailyNews/sub000814.html" >
Grim Prospect for Crewmen Trapped in Russian Sub
</a>
</b></font><br>
<font face=geneva,arial,helvetica size=2>More than 100 Russian crewmen are trapped in a nuclear submarine on the ocean floor
<a href="http://abcnews.go.com/sections/world/DailyNews/sub000814.html" ><img src="http://a4.g.akamaitech.net/7/4/622/000/abcn
<br clear=all>
```

图 1-1 ABC News 网站 2000 年 8 月 14 日头条新闻的标记，使用了表格布局，标题为大号粗体文本；代码没有结构，难以理解

正当 Web 变得一团糟之际，CSS 作为解决方案面世了。CSS 的初衷是把跟 HTML 混在一起的表现性标记提取出来，使其自成体系，达到**结构与表现分离**的目的。这就让有意义的标签或者说语义悄悄返回了 HTML 文档。`font` 之类的表现性标签可以不用了，而表格布局也可以被逐步取代。对大多数网站而言，CSS 都能提升其可访问性和加载速度。不仅如此，CSS 还给 Web 设计和开发人员带来了更多好处：

- 一种专用于控制视觉样式和布局的语言；
- 在同一网站中更易于重用的样式；
- 通过关注点分离得到了良好的代码结构。

关注点分离

关注点分离（separation of concerns）是软件开发行业的一个常见概念。对Web开发而言，关注点分离不仅适用于标记和样式，同样也适用于编写样式的方式。事实上，关注点分离也是确保代码可维护性的一种主要方法。

Unix开发社区有一句话很好地诠释了关注点分离的思想，即“分成小块，松散结合”（small pieces, loosely joined）。其中，每一“小块”都是一个模块，专注于做好一件事。而且，因为这个模块跟其他组件是“松散结合”的，所以它可以方便地在系统的其他部分中重用。Unix中的一“小块”可能是一个字数统计函数，可以应用于传入的任何文本片段。而在Web开发中，这一“小块”可能就是一个商品列表组件，如果能做到“松散结合”，就可以

在一个网站的多个页面或者在同一布局的不同区块中重用。

可以把代码中的这些“小块”想象成积木。每一块积木都很简单，但把很多块积木以不同方式组装起来，就可以创造出无比复杂的东西。第12章还会讨论这个话题，届时会讲一讲怎样以结构化的方式应用这个策略。

1. HTML 和 CSS 的版本

CSS 有很多版本，或者“级别”。了解这些版本产生的背景，有助于了解应该或不应该使用哪些 CSS 特性。万维网联盟（W3C，World Wide Web Consortium）是制定 Web 技术标准的组织，该组织制定的每一个规范要经历几个阶段之后才能成为 W3C 推荐标准。CSS1 是在 1996 年底成为 W3C 推荐标准的，当时只包含字体、颜色和外边距等基本的属性。CSS2 在 1998 年成为推荐标准，增加了浮动和定位等高级特性，此外还有子选择符、相邻选择符和通用选择符等新选择符。

相比之下，CSS3 则采用了完全不同的模式。实际上**不存在**所谓的 CSS3 规范，因为 CSS3 指的是一系列**级别**独立的模块。如果规范模块是对之前 CSS 概念的改进，那就从 3 级（level 3）开始命名。如果不是改进，而是一种全新的技术，那就从 1 级（level 1）开始命名。而我们所提到的 CSS3，则是指所有足够新的 CSS 规范模块，比如 CSS Backgrounds and Borders Level 3、Selectors Level 4 和 CSS Grid Layout Level 1。这种模块化的方式可以让不同的规范有自己的演进速度。有些 3 级规范，比如 CSS Color Level 3，已经成为推荐标准。而另外一些可能还处于候选推荐阶段，很多甚至还处于工作草案阶段。

虽然 CSS3 的制定工作在 CSS2 发布后就开始了，但这些新规范一开始的制定速度很缓慢。为此，W3C 在 2002 年发布了 CSS2 Revision 1。CSS 2.1 修正了 CSS2 中的一些错误，删掉了支持度不高或者并非所有浏览器都实现了的一些特性，总体来说就是把 CSS 规范做了一番清理，好为浏览器实现提供更精准的蓝图。CSS 2.1 在 2011 年 6 月成为推荐标准，此时距离 CSS3 启动已经有 10 多年了。由此可见，标准制定主体和浏览器开发商为了确保相应的特性得以原原本本地实现，需要花多么长的时间。不过，浏览器开发商经常会在标准还处于草案阶段时，就发布一些实验性的实现。这样，等到了候选推荐阶段，相应的实现就已经非常稳定了。换句话说，很多 CSS 特性早在相应模块成为推荐标准前就可以使用了。

HTML 的历史也很复杂。HTML 4.01 在 1999 年成为推荐标准，与此同时 W3C 也把注意力转向了 XHTML 1.0。本来接着要发布 XHTML 1.1，但其严格程度在实践中暴露了无法落地的问题，最终被 Web 开发社区抛弃。于是，这个 Web 主要语言的发展停滞了。

2004 年，有几家公司共同组建了 Web 超文本应用技术工作组（WHATWG，Web Hypertext Application Technology Working Group），并致力于开发新的规范。2006 年，W3C 肯定了它们工作的必要性，并欣然加入该工作组。2009 年，W3C 完全放弃 XHTML，正式接纳 WHATWG 制定的新标准，这就是后来的 HTML5。起初，WHATWG 和 W3C 都基于标准调整自己的工作，但后来它们的关系又变得复杂起来。今天，它们分别在编辑两份标准。WHATWG 那份就叫 HTML，

而 W3C 那份则称为 HTML5。没错，这种分裂确实不好。但万幸的是，这两份标准的内容相当接近，因此只讲 HTML5 是没有问题的。

2. 应该使用哪个版本

设计者和开发者经常问的一个问题就是应该使用 HTML 或 CSS 的哪个版本。这个问题不好回答。虽然规范反映了标准和 Web 技术开发的进度和焦点，但它其实跟设计者和开发者日常的工作关系不大。真正重要的是知道 HTML 和 CSS 的哪些部分已经在浏览器中实现了，以及这些实现是否稳健，有没有 bug。比如，浏览器提供的这些特性是不是实验性特性，使用时需谨慎？或者，这些特性到底靠不靠谱，是不是已经得到了多数浏览器的支持？

今天，使用 CSS 和 HTML 就要了解浏览器对其中特性的支持程度。有时候我们会觉得技术发展太快，必须拼命追赶才不会落伍；有时候我们又会觉得技术发展太慢。本书会随时提示不同 HTML 和 CSS 特性的浏览器支持情况，还会给出在什么情况下可以使用它们的建议。不过显而易见的是，印在纸上的东西注定会过时，所以你得学会自己更新这方面的信息。

要了解浏览器支持情况，推荐几个不错的地方。对于 CSS 属性，可以访问“Can I use”网站（https://caniuse.com）。这个网站可以搜索属性或属性组，结果配有统计信息，显示支持它们的浏览器百分比，包括桌面浏览器和移动浏览器。另一个非常有想法的项目是 https://webplatform.github.io/，是 W3C 和几家浏览器厂商及行业巨头合作搞出来的，目标是收集合并它们所有关于 CSS、HTML、JavaScript API 等支持情况的文档。不过，就跟很多大型项目一样，最终要完成那么庞大的 Web 技术文档的聚合，需要花很长时间。此外，Mozilla 的开发者文档，即 MDN，也是一个非常好的参考。

说到浏览器支持，关键要明白，并非所有浏览器都一样，实际上从来就没有完全一样的浏览器。某些 CSS3 特性只得到了少数浏览器的支持。比如，Internet Explorer 11 和 Safari 6.1 之前的版本都没有正确支持 Flexbox（Flexible Box Layout）。不过，就算需要支持老版本的浏览器，也不意味着不能使用 Flexbox。核心布局上可以不用 Flexbox，但对于某些特定的组件，Flexbox 可能就非常合适。只要为不支持它的浏览器准备好可以接受的后备代码就行了。判断一个人是不是 CSS 大师，很大程度上要看他能否游刃有余地处理向后兼容的代码与未来友好的代码。

1.1.3 渐进增强

平衡向后兼容性与最新的 HTML 和 CSS 特性，涉及一种叫作**渐进增强**（progressive enhancement）的策略。所谓渐进增强，大意就是“首先为最小公分母准备可用的内容，然后再为支持新特性的浏览器添加更多交互优化”。使用渐进增强策略，意味着代码要分层，每一层增强代码都只会在相应特性被支持或被认为适当的情况下应用。听起来有点复杂，而实际上 HTML 和 CSS 的实现已经部分内置了这一策略。

对 HTML 而言，这意味着浏览器在遇到未知元素或属性时并不会报错，而且也不会对页面产生什么影响。比如，可以在页面里使用 HTML5 定义的新 `input` 元素。假设表单中有一个电子